Routing problem under the shared storage policy for unit-load automated storage and retrieval systems with separate input and output points

Tanaka, Shunji; Araki, Mituhiko

Routing problem under the shared storage policy for unit-load automated storage and retrieval systems with separate input and output points

Shunji Tanaka*
Graduate School of Engineering, Kyoto University
Kyotodaigaku-Katsura, Nishikyo-ku
Kyoto City, Kyoto 615–8510, Japan

Mituhiko Araki†
Matsue National College of Technology
14–4 Nishiikuma-Cho
Matsue City, Shimane 690-8518, Japan

In this study the routing problem for unit-load automated storage and retrieval systems (AS/RSs) with separate input and output points is considered under the shared storage policy. This problem is to find an optimal travel route of an S/R (storage and retrieval) machine to process given storage and retrieval requests so that the total travel time is minimized, where the input and output points are possibly separate and the shared storage policy is assumed. We first give two types of formulations as 0-1 integer linear programming problems corresponding to two types of dwell point settings: the dwell point is the input point and the output point. Next, we propose a simple but efficient exact solution algorithm based on the formulations that utilizes a general MILP (Mixed Integer Linear Programming) solver. Then, its efficiency is shown by numerical experiments. Indeed, instances with 400 items (200 for each storage and retrieval) are solved within 100 seconds.

Keywords: unit-load AS/RSs, shared storage policy, separate input and output points, routing problem, 0-1 integer linear programming, exact algorithm.

1 Introduction

Automated storage and retrieval systems (AS/RSs) have become widely used in warehouses and distribution centres. A typical AS/RS consists of several storage racks along parallel aisles and S/R (storage and retrieval) machines. Each S/R machine can move along an aisle to process storage and retrieval requests from/to the input/output point at one end of the aisle. To operate such an AS/RS, several problems should be solved; where storage items should be stored, which items should be processed in each cycle of the S/R machine, where the S/R machine should dwell when no request exists, and so on (van den Berg 1999).

As the methods to determine the position where storage items should be stored, there are two main classes of policies (Goetschalckx and Ratliff (1990), Kulturel *et al.* (1999)):

(1) dedicated storage policy
Storage items are stored in their corresponding rack positions.
(2) shared storage policy
Storage items can be stored in any rack openings.

The shared storage policy often appears in literatures as the following two policies (Hausman *et al.* 1976).

(a) class-baseds torage policy
A storage rack is partitioned into classes or zones, and storage items are stored in their corresponding zones. There are several researches on the storage rack zoning (Rosenblatt and Eynan (1989), Guenov and Raesid (1992), Eynan and Rosenblatt (1994), van den Berg and Gademann (1996), and so on).

*Corresponding author. E-mail: tanaka@kuee.kyoto-u.ac.jp, Tel: +81-75-383-2204, Fax: +81-75-383-2201.
†E-mail: araki@matsue-ct.ac.jp.

(b) random (randomized) storage policy
Storage items are stored in any openings with an equal probability. The shared storage policy itself is sometimes referred to as random (or randomized) storage policy, but we follow the classification by Kulturel *et al.* (1999) for clarity because the very point of the shared storage policy is that storage positions can be determined *arbitrarily*, not randomly.

Under the shared storage policy, there are several heuristics to choose an opening for the next storage item: the closest open location (COL) rule (Hausman *et al.* 1976), the nearest neighbor (NN) and shortest leg (SL) rules (Han *et al.* 1987), the shortest total travel (TT) rule (Lee and Schaefer 1996), and so on.

To process storage and retrieval items, how to route an S/R machine should also be determined in addition to storage positions. In the case of a unit-load S/R machine, items are processed by either a single-command (SC) or dual-command (DC) operation. In a SC operation, one storage item or one retrieval item is processed in one cycle of the S/R machine starting from the I/O point and returning again to the I/O point. On the other hand, in a DC operation one storage item and one retrieval item are processed in one cycle. Therefore, it is better in general to perform a DC operation if there are both storage and retrieval items. In most researches, storage items are assumed to be processed in FCFS (First-Come-First-Served) order because it is often the case that storage items are transported to the AS/RS by a conveyor system linked to the I/O point. As for retrieval sequencing, i.e., how retrieval items should be paired with storage items to perform DC operations, the FCFS rule, the NN, SL and TT rules (these three rules determine the next storage position and the next retrieval item at the same time), and the nearest retrieval (NR) and nearest storage/retrieval (NSR) rules (van den Berg and Gademann (2000); the NSR rule may perform a SC storage operation) among others are known. There is also an attempt to switch such rules dynamically (Yin and Rau 2006). However, only few researches consider exact optimization methods for routing the S/R machine.

In the case of the dedicated storage policy Lee and Schaefer (1997) treated the routing problem of a unit-load S/R machine as a linear assignment problem, which is known to be polynomially solvable. Another research under the dedicated storage policy was done by van den Berg and Gademann (1999). They showed that the problem under FCFS storage sequencing can be formulated as a transportation problem and hence is polynomially solvable even when the input and output points are separate. On the other hand, in the case of the shared storage policy Lee and Schaefer (1996) proposed a solution algorithm by regarding the routing problem as a linear assignment problem with an additional constraint that storage items cannot be stored in the location where a retrieval item occupies before it is retrieved. If an optimal solution of the linear assignment problem does not satisfy the constraint, the second best solution is constructed by the ranking algorithm (Murty 1968). It is continued until a feasible solution or a sufficiently good solution compared to an upper bound is obtained. This algorithm works as an exact algorithm when it is iterated until an optimal solution (a feasible solution or a solution equal to an upper bound) is obtained, and as an efficient near-optimal algorithm when the iteration is terminated before obtaining an optimal solution to reduce computational efforts.

In this study we propose an exact algorithm for a more general routing problem of a unit-load S/R machine. In our problem, possibly separate input and output points are considered under the shared storage policy (see Table 1). One of our main results is that the problem is formulated as two types of 0-1 integer linear programming problems corresponding to two types of dwell point settings: the dwell point is the input point and the output point. It, in general, enables us to solve the problem by utilizing a general MILP (Mixed Integer Linear Programming) solver. Unfortunately, our formulations have exponentially many number of constraints and it is difficult to apply a MILP solver to them directly. To cope with this difficulty, we propose a simple but efficient iterative solution algorithm based on a MILP solver where constraints are taken into account only when they are violated. Then, not only the effectiveness of our algorithm but also the effects of the rack shape and the dwell point setting on the performance of the AS/RS are examined by computational experiments.

2 Problem Description and Solution Components

In this section, we present an explicit problem description and show several components that compose a solution of the problem.

Consider a storage rack and a unit-load S/R machine. The S/R machine operates to process storage and retrieval requests for the rack. These requests arrive at time zero and can be processed in an arbitrary order (or, we can assume that storage requests are processed in FCFS order because they are not distinguished from each other in our problem).

Storage items waiting at the input point should be carried into some openings of the rack. On the other hand, retrieval items should be carried out from the rack to the output point. A position where a retrieval item occupies becomes an opening after it is retrieved. A storage item can be stored in either such an opening or an initial opening. The input and output points are possibly separate. For example, the input point is at one end of an aisle, while the output point is at the other end. The objective in this study is to find an optimal travel route of the S/R machine such that all the storage and retrieval items are processed and the total travel time is minimized.

For this problem, we make the following basic assumptions.

Assumption 2.1 Travel time of the S/R machine satisfies the triangle inequality.

Assumption 2.2 Item pickup/drop time is constant and thus is ignored.

Assumption 2.3 The S/R machine dwells at the input or output point initially and returns there again after all requests are processed.

Assumption 2.4 There are at least one storage request and one retrieval request.

Hereafter, we use the following notation and definitions. The numbers of storage and retrieval items are denoted by $n_{\mathrm{S}} > 0$ and $n_{\mathrm{R}} > 0$, respectively. The number of initial openings is denoted by n_{O} and let $n = n_{\mathrm{R}} + n_{\mathrm{O}}$. A location L_i $(0 \le i \le n+1)$ denotes

$$L_i : \begin{cases} \text{the input point}, & \text{if } i = 0, \\ \text{the } i\text{-th retrieval point}, & \text{if } 1 \le i \le n_{\mathrm{R}}, \\ \text{the } (i - n_{\mathrm{R}})\text{-th initial opening}, & \text{if } n_{\mathrm{R}} + 1 \le i \le n, \\ \text{the output point}, & \text{if } i = n + 1. \end{cases}$$

Here, "i-th retrieval point" means the rack position where the i-th retrieval item occupies. The travel time of the S/R machine from L_i to L_j is denoted by c_{ij}.

From Assumption 2.1, we can assume without loss of optimality that if the S/R machine visits a location except the final dwell point, it always picks up or drops an item there. Therefore, all the possible components (sub-travels and sub-cycles) of an optimal solution are given as follows (See also Figure 1).

(a) Single Storage Command Cycle (SSCC)
The S/R machine picks up a storage item at the input point, moves to an opening, stores the item, and then returns to the input point.

(b) Single Storage Command Travel (SSCT)
The S/R machine picks up a storage item at the input point, moves to an opening, stores the item, and then moves to the output point.

(c) Single Retrieval Command Cycle (SRCC)
The S/R machine starts from the output point, visits a retrieval point, picks up a retrieval item, returns to the output point, and then drops the item.

(d) Single Retrieval Command Travel (SRCT)
The S/R machine starts from the input point, visits a retrieval point, picks up a retrieval item, moves to the output point, and then drops the item.

(e) Dual Command Travel (DCT)
The S/R machine picks up a storage item at the input point, moves to an opening, stores the storage

item, moves to a retrieval point, picks up a retrieval item, moves to the output point, and then drops the retrieval item.

(f) Output point to Input point Travel (OIT)
The S/R machine moves from the output point to the input point.

In the following, these components are denoted by their abbreviations with visited location numbers in parentheses, if necessary. For example, a dual command travel that visits the locations L_3 and L_1 in this order is denoted by DCT(3, 1).

Since an optimal solution consists of these components, it can be expressed by a sequence of components as DCT(3, 1), SRCC(2), OIT, SSCC(1). In this case, the solution corresponds to a cycle $L_0 \to L_3 \to L_1 \to L_4 \to L_2 \to L_4 \to L_0 \to L_1 \to L_0$, where the output point is assumed to be L_4. It is easy to see that a feasible sequence satisfies the following conditions:

- All the components are connected.
- All the storage items are stored and all the retrieval items are retrieved.
- The initial and final positions of the S/R machine are identical (the input or output point).
- Precedence relations are satisfied that item retrieval from a location should precede item storage to that location. In other words, a storage item cannot be stored in the location where a retrieval item occupies until it is retrieved.

We will give mathematical formulations as 0-1 integer linear programming problems for the two dwell point settings separately:

(A) The dwell point of the S/R machine is the input point.
(B) The dwell point of the S/R machine is the output point.

In the next section, only the problem formulation for the dwell point setting (A) is shown. The results for the dwell point setting (B) are almost parallel to those for the dwell point setting (A), and hence the formulation is given in Appendix A.

3 Problem Formulation for Dwell Point Setting (A)

In this section we consider the dwell point setting (A), i.e. the case when the dwell point is the input point, and give a 0-1 integer programming problem formulation for this dwell point setting. For preparation, we show some basic properties.

3.1 *Basic Properties*

To begin with, we show the following property claiming that it is not necessary to consider SSCTs.

Property 3.1 It is not necessary to consider SSCTs in an optimal sequence. Thus, there exists an optimal sequence consisting of SRCCs, SRCTs, SSCCs, DCTs and OITs.

Proof Assume that there exists an optimal sequence containing SSCTs. Then, any subsequence starting from an SSCT in the sequence is given by

SSCT, SRCC, ..., SRCC, OIT, ...,

or,

SSCT, OIT, ...

In the former case, the SSCT and the SRCC just after the SSCT can be combined into a DCT without increasing the total travel time from Assumption 2.1. For example, an SSCT(1) and an SRCC(2) can be combined into a DCT(1, 2). In the latter case, the SSCT and the OIT can be combined into an SSCC without increasing the total travel time. Therefore, we need not consider SSCTs. □

If a feasible sequence satisfies Property 3.1, it also satisfies the following two properties.

Property 3.2 In any feasible sequence without SSCTs, the number of occurrences of OITs is equal to the number of occurrences of SRCTs and DCTs.

Proof It is obvious because the initial and final positions are the input point. □

Property 3.3 At least one SRCT occurs in any feasible sequence without SSCTs if there is no DCT that visits an initial opening (including the case that there is no DCT at all).

Proof It is obvious that an SRCT is necessary if no DCT occurs at all because the initial position of the S/R machine is the input point and it cannot move to the output point without SRCTs, DCTs nor SSCTs. Thus, we assume that at least one DCT occurs, but no DCT visits an initial opening. Such a DCT cannot precede item retrieval that makes the storage point visited in that DCT open. It implies that an SRCT or an SRCC should precede all the DCTs to open at least one storage point. Moreover, an SRCC can be processed only after at least one SRCT or DCT occurs because the S/R machine should move from the input point to the output point to process an SRCC. Therefore, at least one SRCT should occur. □

The last property is on an optimal sequence.

Property 3.4 There exists an optimal sequence such that

- all the SRCCs occur contiguously after the first SRCT (if exists) or DCT (if no SRCT exists),
- all the SSCCs occur contiguously at the end of the sequence.

Proof It is obvious because it is better to retrieve items as early as possible and to store items as late as possible to meet precedence relations. □

3.2 *Problem Formulation*

From the properties shown in the preceding subsection, we formulate the problem as a 0-1 integer linear programming problem.

Let us introduce binary decision variables x_i $(1 \le i \le n)$ and y_{ij} $(0 \le i \le n,\ 1 \le j \le n_{\mathrm{R}},\ i \ne j)$ defined by

$$x_i = \begin{cases} 1, & \text{if } L_i \text{ is used as a storage point,} \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

$$y_{ij} = \begin{cases} 1, & \text{if } i \ne 0 \text{ and a DCT}(i,\, j) \text{ occurs, or, if } i = 0 \text{ and an SRCT}(j) \text{ occurs,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In addition, we define V_{R} by $V_{\mathrm{R}} = \{1, \ldots, n_{\mathrm{R}}\}$ and $\bar{c}_{ij}$ $(0 \le i \le n,\ 1 \le j \le n_{\mathrm{R}},\ i \ne j)$ by

$$\bar{c}_{ij} = \begin{cases} c_{ij} + c_{0,n+1} - c_{0i} - c_{j,n+1}, & \text{if } i \ne 0, \\ c_{0j} + c_{0,n+1} - c_{j,n+1}, & \text{if } i = 0. \end{cases} \tag{3}$$

Then, our problem with the dwell point setting (A) can be formulated as a 0-1 integer linear programming problem (P^{A}):

$$\text{minimize} \quad \sum_{1 \le i \le n_{\mathrm{R}}} 2c_{i,n+1} + \sum_{1 \le i \le n} 2c_{0i}x_i + \sum_{\substack{0 \le i \le n \\ 1 \le j \le n_{\mathrm{R}} \\ i \ne j}} \bar{c}_{ij}y_{ij}, \tag{4}$$

$$\text{subject to} \quad \sum_{1 \le i \le n} x_i = n_{\mathrm{S}}, \tag{5}$$

$$\sum_{1 \le j \le n_{\mathrm{R}}} y_{0j} + \sum_{\substack{n_{\mathrm{R}}+1 \le i \le n \\ 1 \le j \le n_{\mathrm{R}}}} y_{ij} \ge 1, \tag{6}$$

$$\sum_{\substack{0\le i\le n\\ i\neq j}} y_{ij} \le 1 \qquad (1 \le j \le n_{\mathrm{R}}), \tag{7}$$

$$\sum_{\substack{1\le j\le n_{\mathrm{R}}\\ j\neq i}} y_{ij} \le x_i \qquad (1 \le i \le n), \tag{8}$$

$$\sum_{\substack{i,j\in V_{\mathrm{C}}\\ i\neq j}} y_{ij} \le |V_{\mathrm{C}}| - 1 \qquad (V_{\mathrm{C}} \subseteq V_{\mathrm{R}},\ |V_{\mathrm{C}}| \ge 2), \tag{9}$$

$$x_i \in \{0,1\} \qquad (1 \le i \le n), \tag{10}$$

$$y_{ij} \in \{0,1\} \qquad (0 \le i \le n,\ 1 \le j \le n_{\mathrm{R}},\ i \neq j). \tag{11}$$

We will state step by step where the equations come from. First, the objective function (4) is explained. Suppose that $y_{ij} = 0$ holds for any i and j ($0 \le i \le n$, $1 \le j \le n_{\mathrm{R}}$, $i \neq j$), although it cannot happen from Property 3.3. Then, all the storage points (L_i with $x_i = 1$) are visited by SSCCs and all the retrieval points are visited by SRCCs. Thus, the total travel time of these components is given by

$$\sum_{1\le i\le n_{\mathrm{R}}} 2c_{i,n+1} + \sum_{1\le i\le n} 2c_{0i}x_i. \tag{12}$$

If $y_{ij} = 1$ for some i, j ($1 \le i \le n$, $1 \le j \le n_{\mathrm{R}}$, $i \neq j$), an SSCC(i) and an SRCC(j) disappear and a DCT(i, j) appears. Moreover, an OIT occurs from Property 3.2. Thus, the total travel time increases by $\bar{c}_{ij} = c_{ij} + c_{0,n+1} - c_{0i} - c_{j,n+1}$ as shown in Figure 2(a). In the case that $y_{0j} = 1$ holds for some j ($1 \le j \le n_{\mathrm{R}}$), an SRCC($j$) disappears, and an SRCT(j) and an OIT appear. Then, the total travel time increases by $\bar{c}_{0j} = c_{0j} + c_{0,n+1} - c_{j,n+1}$ (Figure 2(b)). Therefore, the objective function is expressed by (4).

Next, the constraints (5)–(9) are explained. The constraint that all the storage items should be processed is given by (5). The constraint (6) corresponds to Property 3.3 since the first and second terms of the lefthand side of (6) respectively denote the number of occurrences of SRCTs and the number of occurrences of DCTs that visit initial openings. The constraint (7) requires that a retrieval point L_j cannot be visited in two or more DCTs (and SRCTs), and the constraint (8) requires that L_i cannot be visited for storage in a DCT unless it is used as a storage point ($x_i = 1$).

The constraint (9) originates from precedence relations between storage and retrieval. More specifically, (9) is necessary and sufficient for the existence of a solution satisfying precedence relations. In other words, there exists a precedence deadlock called tour (Lee and Schaefer 1996) formed by DCTs if and only if (9) is broken for some V_{C}. Here, we will give a brief sketch of what (9) requires by an example. For the detailed proof of the necessity and sufficiency of (9), please refer to Appendix B.

Suppose that $V_{\mathrm{C}} = \{1, 2, 3\}$ and $y_{12} = y_{23} = y_{31} = 1$. Then, (9) is broken for this V_{C} because both the lefthand and righthand sides of (9) are equal to 3. In this case, there are three DCTs corresponding to y_{12}, y_{23} and y_{31}. The DCT(1, 2) should precede the DCT(2, 3) because of the precedence relation between the storage to L_2 in the latter DCT and the retrieval from L_2 in the former DCT. Similarly, the DCT(2, 3) should precede the DCT(3, 1), and the DCT(3, 1) should precede the DCT(1, 2). Therefore, the three DCTs form a tour and there exists no feasible sequence satisfying the precedence relations.

If the binary decision variables x_i and y_{ij} satisfy the constraints (5)–(9), there exists at least one feasible sequence satisfying the conditions of Property 3.4. First, all the DCTs (and OITs) are sequenced so as not to break precedence relations. Such a sequence always exists because there is no tour from the constraint (9). Next, if there exists some SRCTs, they together with OITs are added to the top of the sequence. Then, all the SRCCs are inserted after the first component. Finally, all the SSCCs are added to the end of the sequence.

For example, consider the case when $n_{\mathrm{R}} = n_{\mathrm{S}} = 6$ and $n_{\mathrm{O}} = 2$, and assume that the decision variables x_1, x_2, x_3, x_6, x_7, x_8, y_{01}, y_{12}, y_{26}, y_{64}, y_{73} are one and the others are all zero. It is easy to check that these decision variables satisfy the constraints (5)–(9). First, DCTs and OITs are sequenced. In this case, there are two blocks formed by precedence relations: The first block is DCT(1, 2), OIT, DCT(2, 6), OIT,

DCT(6, 4), OIT, and the second is DCT(7, 3), OIT. Therefore, the following sequence does not break precedence relations:

DCT(1, 2), OIT, DCT(2, 6), OIT, DCT(6, 4), OIT, DCT(7, 3), OIT.

Next, SRCT(1) corresponding to $y_{01} = 1$ together with an OIT is added to the top of the sequence as follows:

SRCT(1), OIT, DCT(1, 2), OIT, DCT(2, 6), OIT, DCT(6, 4), OIT, DCT(7, 3), OIT.

Then, SRCC(5) yielded by $\sum_{\substack{0 \le i \le n \\ i \ne j}} y_{ij} = 0$ for $j = 5$ is inserted after the first component SRCT(1):

SRCT(1), SRCC(5), OIT, DCT(1, 2), OIT, DCT(2, 6), OIT, DCT(6, 4), OIT, DCT(7, 3), OIT.

Finally, SSCC(3) and SSCC(8) yielded by $x_i = 1$, $\sum_{\substack{1 \le j \le n_{\mathrm{R}} \\ j \ne i}} y_{ij} = 0$ for $i = 3, 8$ are added to the end of the sequence, and we obtain a feasible sequence:

SRCT(1), SRCC(5), OIT, DCT(1, 2), OIT, DCT(2, 6), OIT, DCT(6, 4), OIT, DCT(7, 3), OIT, SSCC(3), SSCC(8).

It should be pointed out that the above sequence is not the unique feasible sequence. For example, SRCC(5) can be moved after any DCT, DCT(6, 4) and DCT(7, 3) can be exchanged, and so on. It is because total travel time does not depend on the sequencing order of components, and they can be sequenced arbitrarily as far as they are connected from the initial dwell point to the final dwell point and precedence relations are satisfied.

4 Solution Algorithm

In this section, we give a simple but efficient exact solution algorithm for our problem formulated in the preceding section.

Instead of constructing a specific branch-and-bound or branch-and-cut algorithm, we utilize a general MILP solver to solve the 0-1 integer linear programming problem (P^{A}). However, it is difficult to apply an MILP solver directly to (P^{A}) because the number of constraints (9) increases exponentially as n_{R} increases. To overcome this difficulty, we start from a problem without (9), and these constraints are added only when they are violated. The detailed algorithm for (P^{A}) is given by the following.

0° Denote by ($\mathrm{P}^{\mathrm{A}}_0$) a 0-1 integer linear programming problem defined by (4)–(8), (10) and (11). Let $i := 0$.
1° Solve ($\mathrm{P}^{\mathrm{A}}_i$) by a general MILP solver.
2° Find tours in the current solution. If no tour is found, output the current solution as an optimal solution and terminate.
3° Construct ($\mathrm{P}^{\mathrm{A}}_{i+1}$) by adding (9) corresponding to the tours to ($\mathrm{P}^{\mathrm{A}}_i$). For example, if $y_{12} = y_{23} = y_{31} = 1$ holds in the current solution, the constraint for $V_{\mathrm{C}} = \{1, 2, 3\}$, i.e.

$$y_{12} + y_{13} + y_{21} + y_{23} + y_{31} + y_{32} \le 2 \tag{13}$$

is added.
4° Let $i := i + 1$ and go to 1°.

It is direct to modify the algorithm for (P^{B}), the problem formulated for the dwell point setting (B), and so is omitted.

5 Numerical Experiments

We examine by numerical experiments the effectiveness of our algorithm proposed in the preceding section. We also examine the effects of the rack shape and the dwell point setting on the performance of the AS/RS.

The rack configuration and the performance of the S/R machine used in the experiments follow Bozer and White (1984). Let L and H respectively denote the length and the height of the rack. The horizontal and vertical travel speeds of the S/R machine are given by V_{h} and V_{v}, respectively. Thus, the travel time from (0, 0) to (p, q) is computed by $\max(p/V_{\mathrm{h}}, q/V_{\mathrm{v}})$. The times to travel full length and full height are defined by $t_{\mathrm{h}} := L/V_{\mathrm{h}}$ and $t_{\mathrm{v}} := H/V_{\mathrm{v}}$, respectively.

We introduce two parameters T and b defined by $T := \max(t_{\mathrm{h}}, t_{\mathrm{v}})$ and $b := \min(t_{\mathrm{h}}/T, t_{\mathrm{v}}/T)$. The parameter T specifies the rack scale and we set T to 1 without loss of generality. The parameter b is called "shape factor" that specifies the rack shape. Here, we assume $t_{\mathrm{h}} \geq t_{\mathrm{v}}$ ($t_{\mathrm{h}} = T = 1$ and $t_{\mathrm{v}} = b$).

The input and output points are placed at the both ends of the aisle as shown in Figure 4. We consider the two dwell point settings (A) and (B), where

(A) the dwell point of the S/R machine is the input point,
(B) the dwell point of the S/R machine is the output point.

The numbers of storage and retrieval items n_{S} and n_{R} are chosen as $n_{\mathrm{S}} = n_{\mathrm{R}}$. We generate problem instances by varying the shape factor b, the number of retrieval (storage) items n_{R} (n_{S}), and the number of initial openings n_{O}, as shown in Table 2. For every combination of b, n_{R} and n_{O}, 10 problem instances are generated. The scaled horizontal and vertical coordinates of retrieval points and initial openings are determined by uniform distributions $[0, 1]$ and $[0, b]$, respectively. Computation is performed on a Pentium4 3.4 GHz desktop computer. The solution algorithm is is coded in C, and ILOG CPLEX9.1 is used as an MILP solver.

First, we show the computational times of our algorithm in Tables 3 and 4, where the average (and the maximum in parentheses) computational times to obtain optimal solutions are shown in seconds. From these tables, we can see that it takes longer computational time for the dwell point setting (B) than for the dwell point setting (A). It is because (P^{B}) has more decision variables than (P^{A}). Nonetheless, all the problem instances with $n_{\mathrm{R}} = n_{\mathrm{S}} = 200$ are optimally solved within 100 seconds, and the effectiveness of our solution algorithm is confirmed.

Next, we examine the effect of the dwell point settings on total travel time. Table 5 shows the ratios (%) of the average travel times for the two dwell point settings, where the average travel times for the dwell point setting (A) are taken as bases. From Table 5, we see that (A) dominates (B) in most problem instances when $b = 0.2$. When b is larger ($b = 0.6$, 1.0), there are only small differences between the two. Since (P^{A}) is easier to solve than (P^{B}), (A) seems advantageous over (B) from a practical point of view.

Last, we examine which locations are and are not visited in DCTs. The locations visited in DCTs are plotted in Figure 5, where the shape factor b is set to 1.0 and all the combinations of n_{O} and n_{R} are taken into account. DCTs never visit so-called the single command area (van den Berg and Gademann 1999) when the dwell point is chosen as (A), although it does not hold in such a case that all the initial openings and retrieval points are within this area. On the other hand, the single command area does not seem to exist for (B). However, it is revealed by a further observation that total travel time does not increase even if we forbid DCTs to visit the single command area, at least when $b = 1.0$. Therefore, we can conclude that the single command area exists for both the dwell point settings. The locations that are not visited in DCTs, i.e. the locations visited in SSCCs, SSCTs, SRCCs and SRCTs are also plotted and are shown in Figure 6. Although there is not so distinct a tendency as in Figure 5, we can see that the locations closer to the output point and with lower vertical coordinates are more likely to be visited. These facts would be helpful for constructing good heuristics.

6 Conclusion

In this study we proposed an exact solution algorithm for the routing problem of a unit-load S/R machine with separate input and output points under the shared storage policy. We gave two types of 0-1 integer

linear programming formulations corresponding to the different dwell point settings that the dwell point is the input point and is the output point. Based on these formulations we proposed an exact solution algorithm utilizing a general MILP solver. Numerical experiments showed that our algorithm can solve problem instances efficiently even with 400 items (200 for each storage and retrieval). We also showed that AS/RS performance does not depend much on the two dwell point settings and that the single command area also exists in our problem.

It is left for future research to extend our algorithm for any dwell point settings. It seems also interesting to apply our algorithm as a dynamic optimization method.

Appendix A: Problem Formulation for Dwell Point Setting (B)

Here, we consider the dwell point setting (B), the case when the dwell point is the output point. As in Section 3, we first show basic properties satisfied in this dwell point setting and then give a 0-1 integer linear programming problem formulation of the problem.

A.1 *Basic Properties*

The components of an optimal sequence in the dwell point setting (B) are restricted by the following property.

Property A.1 It is not necessary to consider SRCTs in an optimal sequence. Thus, there exists an optimal sequence consisting of SRCCs, SSCCs, SSCTs, DCTs and OITs.

Proof Assume that there exists an optimal sequence containing SRCTs. Then, any subsequence finishing at an SRCT in the optimal sequence is given by

…, OIT, SSCC, …, SSCC, SRCT,

or,

…, OIT, SRCT.

In the former case, the SRCT and the SSCC just before the SRCT can be combined into a DCT. In the latter case, the SRCT and the OIT can be combined into an SRCC. Therefore, we need not consider SRCTs. □

The following three properties correspond to Properties 3.2, 3.3 and 3.4, respectively.

Property A.2 In any feasible sequence without SRCTs, the number of occurrences of OITs is equal to the number of occurrences of SSCTs and DCTs.

Proof It is obvious because the initial and final positions are the output point. □

Property A.3 At least one SSCT occurs in any feasible sequence without SRCTs if there is no DCT such that the retrieval point visited in the DCT is not used as a storage point by any storage items.

Proof It is obvious that an SSCT is necessary if no DCT occurs at all because the final position of the S/R machine is the output point and it cannot return to the output point without SSCTs, DCTs nor SRCTs. Thus, we assume that at least one DCT occurs and that all the retrieval points visited in DCTs are used as storage points in other components. It follows that the last DCT should precede an SSCT or an SSCC visiting the retrieval point in that DCT as a storage point. Moreover, an SSCC should precede at least one SSCT or DCT because the S/R machine is located at the input point when the SSCC is finished, whereas the final position is the output point. Therefore, at least one SSCT occurs. □

Property A.4 There exists an optimal sequence such that

- all the SRCCs occur contiguously from the beginning of the sequence,
- the last component of the sequence is an SSCT, if at least one SSCT occurs.

Proof The first part is obvious because it is better to retrieve items as early as possible to meet precedence relations. The second part can be shown by noting that the occurrence of an SSCT leads to the existence of a subsequence OIT, SSCC, ..., SSCC, SSCT. This subsequence can be moved to the end of the sequence without breaking precedence relations. □

A.2 *Problem Formulation*

To formulate the problem with the dwell point setting (B), we introduce binary decision variables $\widetilde{y}_{ij}$ $(1 \le i \le n,\ 0 \le j \le n_{\mathrm{R}},\ i \ne j)$ instead of y_{ij}.

$$\widetilde{y}_{ij} = \begin{cases} 1, & \text{if } j \ne 0 \text{ and a DCT}(i,\ j) \text{ occurs, or, if } j = 0 \text{ and an SSCT}(i) \text{ occurs,} \\ 0, & \text{otherwise.} \end{cases} \tag{A1}$$

In addition, we introduce auxiliary binary variables z_i $(1 \le i \le n_{\mathrm{R}})$ corresponding to Property A.3 such that

$$z_i = \begin{cases} 1, & \text{if a DCT}(\bullet,\ i) \text{ occurs and } L_i \text{ is not used as a storage point,} \\ 0, & \text{otherwise.} \end{cases} \tag{A2}$$

We also define $\widetilde{c}_{ij}$ $(1 \le i \le n,\ 0 \le j \le n_{\mathrm{R}},\ i \ne j)$ instead of $\bar{c}_{ij}$ by

$$\widetilde{c}_{ij} = \begin{cases} c_{ij} + c_{0,n+1} - c_{0i} - c_{j,n+1}, & \text{if } j \ne 0, \\ c_{i,n+1} + c_{0,n+1} - c_{0i}, & \text{if } j = 0. \end{cases} \tag{A3}$$

Then, our problem with the dwell point setting (B) can be formulated as a 0-1 integer linear programming problem (P^{B}):

$$\text{minimize} \quad \sum_{1 \le i \le n_{\mathrm{R}}} 2c_{i,n+1} + \sum_{1 \le i \le n} 2c_{0i}x_i + \sum_{\substack{1 \le i \le n \\ 0 \le j \le n_{\mathrm{R}} \\ i \ne j}} \widetilde{c}_{ij}\widetilde{y}_{ij}, \tag{A4}$$

$$\text{subject to} \quad \sum_{1 \le i \le n} x_i = n_{\mathrm{S}}, \tag{A5}$$

$$\sum_{1 \le i \le n} \widetilde{y}_{i0} + \sum_{1 \le i \le n_{\mathrm{R}}} z_i \ge 1, \tag{A6}$$

$$2z_j \le 1 - x_j + \sum_{\substack{1 \le i \le n \\ i \ne j}} \widetilde{y}_{ij} \quad (1 \le j \le n_{\mathrm{R}}), \tag{A7}$$

$$\sum_{\substack{1 \le i \le n \\ i \ne j}} \widetilde{y}_{ij} \le 1 \quad (1 \le j \le n_{\mathrm{R}}), \tag{A8}$$

$$\sum_{\substack{0 \le j \le n_{\mathrm{R}} \\ j \ne i}} \widetilde{y}_{ij} \le x_i \quad (1 \le i \le n), \tag{A9}$$

$$\sum_{\substack{i,j \in V_{\mathrm{C}} \\ i \ne j}} \widetilde{y}_{ij} \le |V_{\mathrm{C}}| - 1 \quad (V_{\mathrm{C}} \subseteq V_{\mathrm{R}},\ |V_{\mathrm{C}}| \ge 2), \tag{A10}$$

$$x_i \in \{0, 1\} \quad (1 \le i \le n), \tag{A11}$$

$$\widetilde{y}_{ij} \in \{0, 1\} \quad (1 \le i \le n,\ 0 \le j \le n_{\mathrm{R}},\ i \ne j), \tag{A12}$$

$$z_i \in \{0, 1\} \quad (1 \le i \le n_{\mathrm{R}}). \tag{A13}$$

Here, we will explain the meanings of the equations by focusing on the differences between (P^A) and (P^B).

The coefficient $\widetilde{c}_{i0}$ of $\widetilde{y}_{i0}$ in the objective function (A4) is given by (A3) because if $\widetilde{y}_{i0}$ is equal to 1, an SSCT(i) and an OIT occur instead of an SSCC(i). Figure 3 shows this situation. The constraints (A6) and (A7) correspond to Property A.3. The first term in the lefthand side of (A6) denotes the number of occurrences of SSCTs, and the second term denotes the number of occurrences of DCTs satisfying the condition of Property A.3, i.e. the visited retrieval points are not used as storage points by any storage items. Therefore, (A6) requires that at least one SSCT occurs if no DCT satisfies the condition. In addition, the righthand side of (A7) is equal to 2 only when L_j is not used as a storage point ($x_j = 0$) and a DCT($\bullet$, j) occurs. It follows that z_j can be 1 only when the condition is satisfied and it is consistent with the definition of z_j.

If the decision variables x_i, $\widetilde{y}_{ij}$, and z_j satisfy the constraints (A5)–(A13), there exists at least one feasible sequence satisfying the conditions of Property A.4. Such a sequence is given by

SRCC, ..., SRCC, OIT, DCT, OIT, DCT, ..., OIT, DCT, OIT, SSCC, ..., SSCC, SSCT

if some SSCT occurs, or,

SRCC, ..., SRCC, OIT, DCT, OIT, DCT, ..., OIT, DCT, OIT, SSCC, ..., SSCC, DCT

if no SSCT occurs.

Appendix B: Proof of the Necessity and Sufficiency of (9) for Tour Elimination

The sufficiency part is obvious because if there exists a tour, $y_{i_1 i_2} = y_{i_2 i_3} = \cdots = y_{i_{j-1} i_j} = y_{i_j i_1} = 1$ holds for i_1, i_2, ..., i_j corresponding to the tour, and (9) is broken for $V_C = \{i_1, i_2, \ldots, i_j\}$. To show the necessity part, we first note that for any $V_C \subseteq V_R$,

$$\sum_{\substack{i,j \in V_C \\ i \neq j}} y_{ij} \leq |V_C| \tag{B1}$$

holds from (7) and (8). Therefore, if V_C breaks (9),

$$\sum_{\substack{i,j \in V_C \\ i \neq j}} y_{ij} = |V_C| \tag{B2}$$

is satisfied.

Let us suppose that (9) is broken for some V_C. Then, we can assume without loss of generality that V_C has the minimal cardinality among those breaking (9). Under this assumption a vertex set V_C and an arc set E_C compose a connected directed graph (V_C, E_C), where E_C consists of arcs $i \to j$ corresponding to $y_{ij} = 1$ ($i, j \in V_C$, $i \neq j$). If this graph is disconnected, (V_C, E_C) can be decomposed into k maximal connected subgraphs (V_C^i, E_C^i) ($1 \leq i \leq k$). From (B1) and (B2), these subgraphs satisfy $|V_C^i| > 1$ and $|E_C^i| = |V_C^i|$. It follows that V_C^i breaks (9) and the assumption that V_C has the minimal cardinality is contradicted. Therefore, the directed graph (V_C, E_C) has the following properties:

- Connected.
- $|E_C| = |V_C|$ holds (from (B2)).
- Both the in-degree and out-degree of any vertex are less than or equal to 1 (from (7) and (8)).

These properties imply that (V_C, E_C) has a directed Hamiltonian cycle. To summarize, there exists a tour if (9) is broken for some V_C, and the necessity part is proved by contradiction.

REFERENCES

Bozer, Y. A., and White, J. A., Travel-time models for automated storage/retrieval systems, *IIE Transactions*, 1984, **16**, 329–338.

Eynan, A. and Rosenblatt, M. J., Establishing zones in single-command class-based rectangular AS/RS, *IIE Transactions*, 1994, **26**, 38–46.

Goetschalckx, M. and Ratliff, H. D., Shared storage policies based on the duration stay of unit loads, *Management Science*, 1990, **36**, 1120–1132.

Guenov, M. and Raeside, R., Zone shapes in class based storage and multicommand order picking when storage/retrieval machines are used, *European Journal of Operational Research*, 1992, **58**, 37–47.

Han, M.-H., McGinnis, L. F., Shieh, J. S., and White, J. A., On sequencing retrievals in an automated storage/retrieval system, *IIE Transactions*, 1987, **19**, 56–66.

Hausman, W. H., Schwarz, L. B., Graves, S. C., Optimal storage assignment in automatic warehousing systems, *Management Science*, 1976, **22**, 629–638.

Kulturel, S., Ozdemirel, N. E., Sepil, C. and Bozkurt, Z., Experimental investigation of shared storage assignment policies in automated storage/retrieval systems, *IIE Transactions*, 1999, **31**, 739–749.

Lee, H. F., and Schaefer, S. K., Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings, *International Journal of Production Research*, 1996, **34**, 2943–2962.

Lee, H. F., and Schaefer, S. K., Sequencing methods for automated storage and retrieval systems with dedicated storage, *Computers & Industrial Engineering*, 1997, **32**, 351–362.

Murty, K. G., An algorithm for ranking all the assignments in order of increasing cost, *Operations Research*, 1968, **16**, 682–687.

Rosenblatt, M. J. and Eynan, A., Deriving the optimal boundaries for class-based automatic storage/retrieval systems *Management Science*, 1989, **35**, 1519–1524.

van den Berg, J. P., Class-based storage allocation in a single-command warehouse with space requirement constraints, *International Journal of Industrial Engineering*, 1996, **3**, 21–28.

van den Berg, J. P., and Gademann, N., Optimal routing in an automated storage/retrieval system with dedicated storage, *IIE Transactions*, 1999, **31**, 407–415.

van den Berg, J. P., A literature survey on planning and control of warehousing systems, *IIE Transactions*, 1999, **31**, 751–762.

van den Berg, J. P. and Gademann, A.J.R.M., Simulation study of an automated storage/retrieval system, *International Journal of Production Research*, 2000, **38**, 1339-1356.

Yin, Y.-L. and Rau, H., Dynamic selection of sequencing rules for a class-based unit-load automated storage and retrieval system, *International Journal of Advanced Manufacturing Technology*, 2006, **30**, 340–356.

Table 1. Relations among previous studies and our study

storage policy	input/output point	
	common	separate
dedicated	Lee and Schaefer (1997)	van den Berg and Gademann (1999)
shared	Lee and Schaefer (1996)	**this study**

Table 2. Parameter settings

Parameter	Setting
b	0.2, 0.6, 1.0
n_{O}	0, 50, 100, 150, 200
n_{R} (n_{S})	50, 100, 150, 200

Table 3. Average and maximum computational times for dwell point setting (A)

(a) $b = 0.2$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.012s	0.044s	0.095s	0.127s	0.182s
	(0.020s)	(0.069s)	(0.154s)	(0.129s)	(0.319s)
100	0.046s	0.157s	0.299s	0.380s	0.492s
	(0.063s)	(0.259s)	(0.574s)	(0.833s)	(0.761s)
150	0.128s	0.302s	0.676s	0.818s	0.991s
	(0.177s)	(0.615s)	(1.389s)	(1.474s)	(2.136s)
200	0.232s	0.685s	1.168s	1.222s	1.858s
	(0.466s)	(1.636s)	(2.074s)	(1.524s)	(2.919s)

(b) $b = 0.6$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.022s	0.068s	0.102s	0.172s	0.212s
	(0.044s)	(0.125s)	(0.182s)	(0.278s)	(0.369s)
100	0.123s	0.321s	0.463s	0.630s	0.773s
	(0.188s)	(0.662s)	(0.917s)	(0.948s)	(1.287s)
150	0.382s	0.933s	1.404s	1.427s	1.241s
	(0.843s)	(1.853s)	(2.580s)	(2.949s)	(2.188s)
200	0.773s	2.137s	2.446s	3.261s	3.014s
	(0.967s)	(3.893s)	(3.714s)	(9.709s)	(3.899s)

(c) $b = 1.0$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.080s	0.119s	0.150s	0.187s	0.303s
	(0.182s)	(0.179s)	(0.363s)	(0.346s)	(0.458s)
100	0.472s	0.815s	0.623s	0.762s	0.797s
	(1.231s)	(2.080s)	(1.012s)	(0.988s)	(1.233s)
150	2.021s	2.515s	3.078s	2.120s	1.956s
	(4.671s)	(4.889s)	(4.527s)	(5.619s)	(3.289s)
200	7.342s	5.126s	7.050s	4.857s	4.098s
	(15.429s)	(7.524s)	(24.448s)	(10.654s)	(8.408s)

Table 4. Average and maximum computational times for dwell point setting (B)

(a) $b = 0.2$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.208s	0.356s	0.455s	0.482s	0.558s
	(0.348s)	(0.675s)	(0.873s)	(0.717s)	(0.871s)
100	2.113s	3.084s	2.661s	2.918s	2.915s
	(3.149s)	(5.971s)	(4.687s)	(5.724s)	(4.594s)
150	8.844s	8.258s	11.028s	10.023s	9.954s
	(13.948s)	(20.510s)	(28.086s)	(23.508s)	(26.137s)
200	21.172s	29.174s	26.914s	20.983s	25.539s
	(45.558s)	(88.316s)	(54.671s)	(30.308s)	(52.117s)

(b) $b = 0.6$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.333s	0.234s	0.136s	0.204s	0.232s
	(0.681s)	(0.785s)	(0.256s)	(0.322s)	(0.405s)
100	3.127s	1.517s	0.737s	0.903s	0.907s
	(4.442s)	(5.150s)	(1.529s)	(1.390s)	(1.637s)
150	12.104s	2.846s	2.730s	2.481s	1.928s
	(25.294s)	(5.858s)	(4.997s)	(5.238s)	(3.763s)
200	27.764s	7.744s	5.315s	6.946s	5.331s
	(33.849s)	(17.956s)	(8.365s)	(19.889s)	(7.200s)

(c) $b = 1.0$

n_{R}	n_{O}				
	0	50	100	150	200
50	0.609s	0.188s	0.171s	0.202s	0.355s
	(1.073s)	(0.325s)	(0.429s)	(0.349s)	(0.538s)
100	4.780s	1.419s	0.863s	0.981s	0.973s
	(9.950s)	(3.350s)	(1.776s)	(1.341s)	(1.731s)
150	19.446s	4.865s	4.718s	3.280s	2.914s
	(34.631s)	(9.527s)	(7.489s)	(7.297s)	(5.352s)
200	61.791s	11.454s	13.355s	8.164s	6.601s
	(98.974s)	(15.367s)	(47.574s)	(13.840s)	(14.107s)

Table 5. Travel time ratios between (A) and (B)

(a) $b = 0.2$

n_{R}	n_{O} 0	50	100	150	200
50	100.006	100.051	100.042	100.047	100.034
100	99.997	100.019	100.019	100.012	100.008
150	100.000	100.008	100.009	100.007	100.010
200	100.006	100.005	100.006	100.005	100.006

(b) $b = 0.6$

n_{R}	n_{O} 0	50	100	150	200
50	99.981	100.014	100.000	100.000	100.000
100	100.010	100.002	100.000	100.000	100.000
150	100.012	100.000	100.000	100.000	100.000
200	100.008	100.000	100.000	100.000	100.000

(c) $b = 1.0$

n_{R}	n_{O} 0	50	100	150	200
50	99.980	100.000	100.000	100.000	100.000
100	99.998	100.000	100.000	100.000	100.000
150	99.981	100.000	100.000	100.000	100.000
200	100.000	100.000	100.000	100.000	100.000

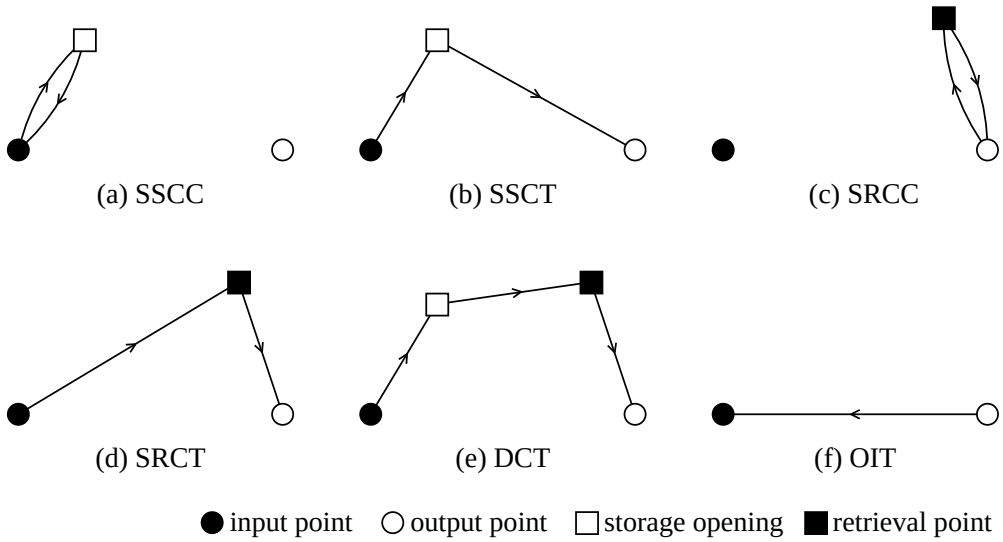
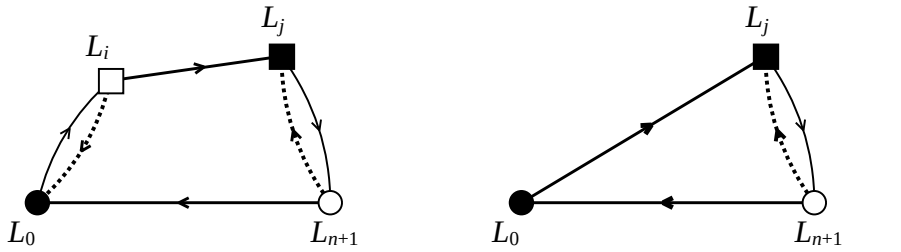


Figure 1. Components of a solution



(a) A DCT and an OIT when $y_{ij} = 1$ (b) An SRCT and an OIT when $y_{0j} = 1$

Figure 2. Relations between y_{ij} and solution components

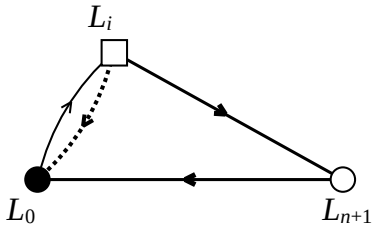


Figure 3. An SSCT and an OIT when $\widetilde{y}_{i0} = 1$

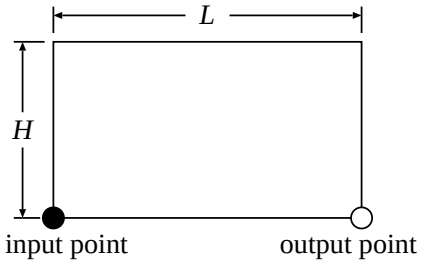
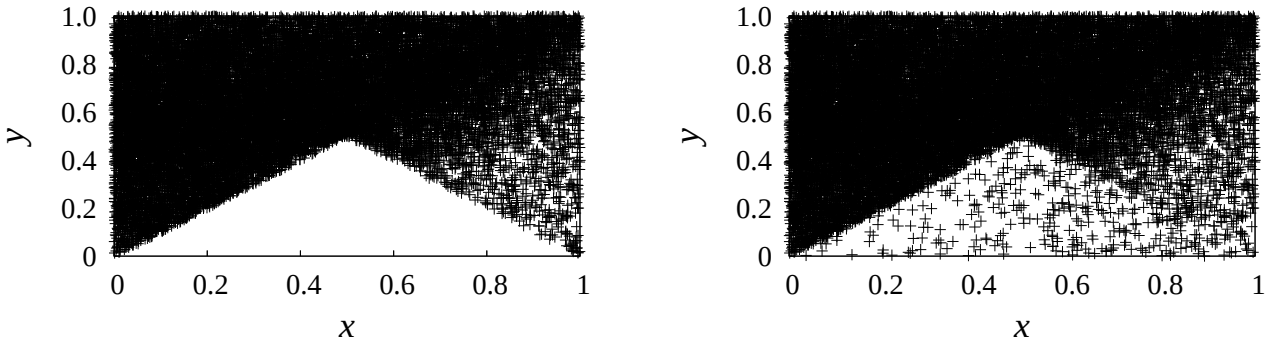


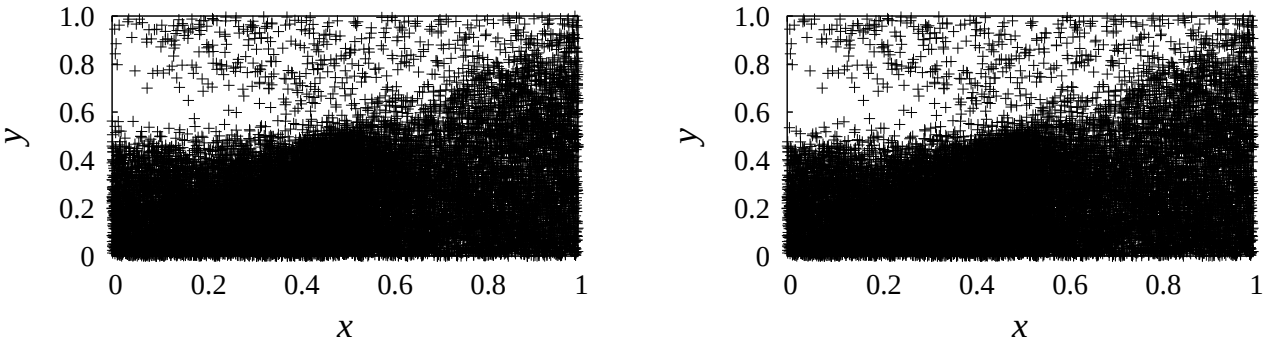
Figure 4. Input and output points



(a) dwell point setting (A)

(b) dwell point setting (B)

Figure 5. Locations visited in DCTs ($b = 1.0$)



(a) dwell point setting (A)

(b) dwell point setting (B)

Figure 6. Locations visited in SSCCs, SSCTs, SRCCs and SRCTs ($b = 1.0$)